[0032]

In Fig. 4, the session is initialized (400), and the session ID related to the current session is acquired from a launcher (406). If the session is not valid (i.e., user is not authenticated), it is obtained from the Java applet or other suitable measures are taken (402). Once the session starts, the Java applet can acquire the login name of the user (408) if the login is usable (404). The Java applet can retrieve the list of roles of the user related to the relevant user only when the login is not unusable (410). The Java applet can also execute an access test in time of execution (412). This test enables a finer access control than those visually recognized through GUI.

[0033]

In Fig. 5, the CGI program tests whether or not the user is already logged in if the secured variable is true (500). If the user is not yet logged in, the secured argument is set to true, OVwwwInit is called out, and a login page is presented to the user. After the user successfully logs in, the CGI program is again called out. Similar to the Java applet of Fig. 4, if the session is not valid, it is obtained from the CGI program or other suitable measures are taken (502). If the session is valid, the session ID is acquired (504). If the login is usable (406), the CGI retrieves the user name (508). The CGI program retrieves the roles of the user (510) and examines whether the user belongs to a specific role (512). The CGI can also perform access test in time of execution (514)

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号
特開2000－207363
（P2000－207363A）
(43)公開日　平成12年７月28日(2000.7.28)

(51)Int.Cl.'	識別記号	FI		テーマコード'(参考)
G06F 15/00	330	G06F 15/00	330B	
H04L 9/32		H04L 9/00	675Z	

審査請求　未請求　請求項の数1　OL　（全 16 頁）

(21)出願番号　特願平11－332208

(22)出願日　平成11年11月24日(1999.11.24)

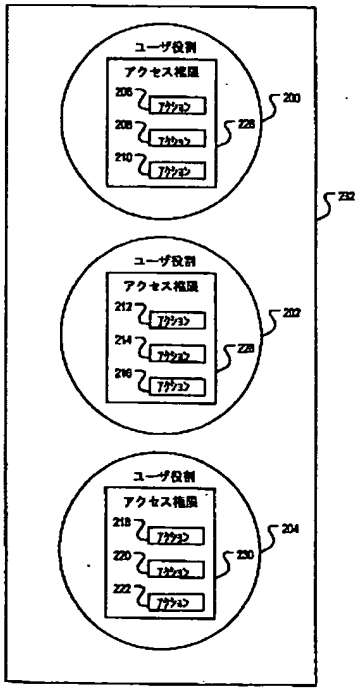
(31)優先権主張番号　09／227124
(32)優先日　平成11年１月７日(1999.1.7)
(33)優先権主張国　米国（US）

(71)出願人　398038580
ヒューレット・パッカード・カンパニー
HEWLETT－PACKARD COMPANY
アメリカ合衆国カリフォルニア州パロアルト　ハノーバー・ストリート　3000
(72)発明者　ローレンス・エム・ピソゥ
アメリカ合衆国80526コロラド州フォート・コリンズ、ガレット・ドライブ　2901
(74)代理人　100081721
弁理士　岡田　次生

最終頁に続く

(54)【発明の名称】　ユーザ・アクセス制御装置

(57)【要約】
【課題】ネットワーク環境においてアクセス権限の過大付与も過小付与もないようにユーザのアクセスを制御する。
【解決手段】"ユーザ役割"を使用してアクセス制御を行う。ユーザ役割は、ユーザの機能が何であるかを決定し、操作上の責任領域を定義する。ユーザ役割によるアクセス制御は、オブジェクトに対してどのようなアクションを取ることができるかを決定するユーザ役割およびアクセス権限を含む。ユーザのユーザ役割が特定のオブジェクトに対して特定のアクションをとる権限をユーザに付与するアクセス権限をユーザに与えていれば、ユーザはそのアクションを実行することができる。



1

【特許請求の範囲】
【請求項１】コンピュータ・オブジェクトに対するユーザのアクセスを制御する装置であって、
１つまたは複数のコンピュータ読み取り可能記憶媒体と、
上記１つまたは複数のコンピュータ読み取り可能記憶媒体に所在して１つまたは複数のユーザ役割を作成するプログラム・コードと、
を備え、
上記１つまたは複数のユーザ役割の各々がユーザ・セットに関連づけられたアクセス権限セットを含み、
上記アクセス権限の各々が上記ユーザ・セットの中のユーザによって実行される上記コンピュータ・オブジェクトに対するアクションを記述する、
ユーザ・アクセス制御装置。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、システム・セキュリティ管理の分野に関するもので、特に、ユーザ役割の使用を通してネットワーク環境における管理およびセキュリティの容易さを推進するようにユーザ・アクセス制御を実施する方法に関するものである。
【０００２】
【従来の技術】ネットワーク環境における管理(ユーザはジョブを実行するため必要なものだけを見る)およびセキュリティ(ユーザは認可されている動作だけを実行することができる)の容易さを推進するためには、ユーザがシステム上で行うことができることを制御することができることが重要である。異なるユーザが異なるレベルの特権を割り当てられネットワークの異なる部分を管理することができるようにすることによってこのような目標を達成するため、ユーザが管理することができるオブジェクトおよびユーザが実行することができる操作を定義するユーザ・アクセス制御が広く使われている。
【０００３】従来技術の製品におけるユーザ・アクセス制御はドメインによって定義されてきた。ドメインという用語の用途は種々あるが、非公式には、共通の管理、位置または規格を持つ一群のオブジェクトを記述するため使用されている。分散環境においては、コンポーネントまたは人々をある単位(またはドメイン)にグループ化することは非常に重要である。なぜならば、そのような環境のサイズおよび複雑さが、管理の目的のため各コンポーネントまたはユーザを１つの独立した主体として扱うことを不可能にするからである。従って、環境を関心および責任領域に細分化することによって管理環境のサイズおよび複雑さを減少させるため、ドメインが使用される。ユーザ・アクセス制御の文脈においては、１つのドメインは、単一の管理ポリシーに左右される資源、人々、特権またはプロセスのようなオブジェクトのセットに属することができる。グループにおけるエレメントが

2

単一ポリシーによって管理されることができるようにドメインはグループ化されなければならない。例をあげれば、ある１つのユーザ・ドメインは、ユーザが何にアクセスすることができるかという単一ポリシーに従う給料支払業務に責任のある一組のユーザであり、オブジェクト・ドメインは同じアクセス権に従う一組の給料支払ファイルである。ドメインのメンバが単一の実体またはドメインであることができ、同じ実体／ドメインが１つまたは複数のドメインに属することもできる。
【０００４】代表的なユーザ・アクセス制御実施形態においては、１つまたは複数のオブジェクト・ドメインに関連づけられる１つまたは複数のユーザ・ドメインがある。ユーザ・ドメインがオブジェクト・ドメインに関して何ができるかを指定するアクセス規則によって、この関連づけがなされる。基本的アクセス規則は、ユーザ・ドメイン、オブジェクト・ドメイン、オブジェクト・ドメインの操作セットを指定する。要求を行うユーザが規則のユーザ・ドメインの中にあり、要求のオブジェクトが規制のオブジェクト・ドメインの中にあり、操作名がオブジェクト・ドメインの操作セットの中にあれば、アクセスが許される。要求におけるユーザ、オブジェクトまたは操作がいずれの既存のアクセス規則にもあてはまらないならば、新しいアクセスが定義される。
【０００５】例えば、アクセス規則{Payroll, Payroll_Files, {read, write}}は、Payroll(給料支払)ユーザ・ドメインにおけるユーザのすべてがPayroll_Files(給料支払ファイル)というオブジェクト・ドメインのオブジェクトにread/write(読み書き)することができることを指定する。異なるユーザがPayrollドメインの範囲内の異なるサブセットの操作を必要とすれば、複数のアクセス規制が作成されなければならない。従って、もしも、Payrollユーザ・ドメインが Payroll_Staff(給料支払スタッフ)ユーザ・ドメインおよびPayroll_Mgr(給料支払管理者)ユーザ・ドメイン(ドメイン・メンバはその他のドメインでもよい)を含み、Payroll_StaffがPayroll_Filesの読み取りだけを許され、一方、Payroll_Mgrが読み書きを許されたとすれば、Payroll_Staff{Payroll_Staff,Payroll_Files,read}という１つのアクセス規則が作成されると共に、別のPayroll_Mgr{Payroll_Mgr,Payroll_File,{read,write}}という規則が作成されるであろう。
【０００６】
【発明が解決しようとする課題】ユーザ・アクセス制御の既存のモデルに関わる１つの問題は、権限がしばしば過大付与または過小付与されることである。権限の過大付与および過小付与はオブジェクト管理およびセキュリティ制御の目標から乖離する。権限の過大付与はシステムをセキュリティ違反に対して脆弱にする。例えば、ObjectDomain1が{read,write,create}というアクセスを持つと定義され、アクセス規則がObjectDomain1{UserDoma

in1，ObjectDomain1，read}に対するreadアクセスをUserDomain1に、ObjectDomain1{UserDomain2，ObjectDomain1，{readwrite}}に対するread/writeアクセスをUserDomain2に、ObjectDomain1{UserDomain3，ObjectDomain1，{read,write,create}}に対するread/write/createアクセスをUserDomain3にそれぞれ付与すると仮定する。更に、例えば一人のユーザが複数の役割を引き受けるか別の従業員の代役をする場合がそうであるように、User1が事務作業の目的からUserDomain1のメンバであり、管理の目的からUserDomain2のメンバであり、運用の目的からUserDomain3のメンバであると仮定する。User1は、システムにサインオンする時、UserDomain1、UserDomain2およびUserDomain3のメンバと認識される。その結果、User1の通常の業務が純粋に事務作業であり、UserDomain1のアクセス権を必要とするだけであるのに、そのユーザ・ドメイン・メンバ権のため、そのユーザは、その仕事のために必要とされる以上の権限であるUserDomain2およびUserDomain3のアクセス権を持つという過剰付与が起きる可能性がある。

【０００７】権限の過小付与は、必要な権限を維持するため貴重な時間と資源を消費する。厳格なセキュリティが存在し、ユーザには必要に応じて権限が与えられるという場合に権限の過小付与が発生する。例えば、User1は、通常は事務員の役割を引き受けているので、ObjectDomain1へのアクセスのためUserDomain1に割り当てられる。User1が一時的に管理者の役割を引き受ける必要があるとすると、User1にUserDomain2の管理の権限を与えるアクセス規則が作成される必要がある。同じように、もしもUser1が一時的に運用従業員の役割を引き受ける必要があるとすれば、その目的のためのアクセス規則が作成されなければならない。User1の一時的任務が解かれると、そのアクセス規則を削除するため時間をさくか、さもなければ、システムは権限の過大付与にさらされる。

【０００８】既存モデルのユーザ・アクセス制御に関わる別の問題は、同じプラットホーム上の複数のアプリケーションでユーザ・アクセス制御が実施される場合に発生する。ユーザ・アクセス制御機能性に対するニーズが非常に大きかったので、アプリケーションは、アプリケーションが存在するプラットホーム上でセキュリティ機能を実施することが必要であるとみなした。例えば、Domain1が操作セット{Read,Write,Create}によって定義され、プラットホームが管理ポリシー{UserGroup1、Domain1、read}を定義し(但しUserGroup1のメンバがreadアクセス権を持つ)、Application1が独立して管理ポリシー{UserGroup1，Domain1，{read,write}}を定義する。UserGroup1のメンバがアプリケーションにサインオンする時Domain1へのread/writeアクセス権を持つと仮定する。

【０００９】第１の問題は、UserGroup1に属し、Domain1のオブジェクトにアクセスするユーザがプラットホームとアプリケーションの間で矛盾した権限の内容を与えられるという点である。なぜならば、プラットホームに従えばユーザはDomain1のオブジェクトへのreadアクセス権を持ち、アプリケーションにおいては同じオブジェクトへのread/writeアクセス権を持つからである。更に、このモデルは、アプリケーション統合すなわち２つのアプリケーションがドメインを越えてオブジェクトを共有する能力に問題を与える。例えば、Application1がプラットホーム{UserGroup1，Domain1，{Read}}と同じ管理ポリシーを定義するとすれば、Application1のユーザは明らかにDomain1のオブジェクトへのwriteアクセス権を持つべきではないのに、Application2に関して独立して作成される管理ポリシーのためにDomain1のオブジェクトへのwriteアクセス権を持つというセキュリティ違反が発生する可能性がある。その結果、アプリケーション統合は実施が一層困難となる。

【００１０】従って、ユーザ・アクセス制御モデルがユーザの特定のジョブに合う権限を付与し、アプリケーション整合性およびアプリケーション統合を促進することができる手段が必要とされている。

【００１１】

【課題を解決するための手段】課題を解決するため本発明が提供するコンピュータ・オブジェクトに対するユーザ・アクセスを制御する装置は、１つまたは複数のコンピュータ読み取り可能記憶媒体、および、上記１つまたは複数のコンピュータ読み取り可能記憶媒体に所在して１つまたは複数のユーザ役割を作成するプログラム・コードを備え、上記１つまたは複数のユーザ役割の各々がユーザ・セットに関連づけられたアクセス権限セットを含み、上記アクセス権限の各々が上記ユーザ・セットの中のユーザによって実行される上記コンピュータ・オブジェクトに対するアクションを記述するように構成される。

【００１２】本発明は、ユーザ・アクセス制御に対する一層柔軟で強力な手法を実施するためユーザ役割という概念を使用する。ドメイン・モデルが１つまたは複数の操作セット／オブジェクト・ドメインのペアに対してユーザが持つアクセス権に基づいて１つのオブジェクトへのアクセスを決定するのに対して、本発明は、ユーザがメンバである１つまたは複数のユーザ役割に関して定義される権利に従ってアクセス権を決定する。本明細書において、"ユーザ"は、ユーザ認証のために使用されるユニークなＩＤによって識別される。また、"ユーザ"は、人物、または、特定の要求に関してユーザ役割を継承するサーバのいずれかを意味する。ユーザは、１つまたは複数のユーザ役割のメンバであることができ、ユーザがログインする時１つまたは複数のユーザ役割の特徴を継承することができる。ユーザ役割は、そのユーザ役割のメンバであるユーザによって実行されることができるア

クションのセットである。(ユーザ役割は、また、他のユーザ役割によっても定義されることができる)。ユーザ役割は、どのアクションがユーザ役割に属しているかを決定するアクセス権限をユーザに与える。ユーザが、自分がメンバである１つまたは複数のユーザ役割にログインすると、ユーザは、ユーザ許可を与えるそのようなユーザ役割のアクションに対するアクセス権を得て、それらのアクションを行う。同じように、サーバは、初期的にユーザに関連づけられるかまたは特定の要求に関してユーザの個性を継承することによって、アクセス権を得ることができ、従って、認証されたアクションに関してフィルタの役を果たすことができる。

【００１３】ユーザ役割によるユーザ・アクセス制御の実施は、ネットワーク環境における容易な管理およびセキュリティという目標を達成する。一般的に、ユーザ・アクセス制御は、混乱を減少し、ユーザがその仕事を行うために必要とするものだけをユーザが見ることを可能にすることによってこのような目標を推進する。複数のアプリケーションが１つのプラットホーム上に導入され、オブジェクト・アクセスに対する断片的な制御が混同、エラーおよびセキュリティ違反につながることがあり得る場合にこれは特に重要である。ユーザ役割は、複数のアプリケーションの間で共有されるオブジェクト上で実行されることができるアクションを制御する管理性／セキュリティのレイヤを注入することによって混同を排除する。例えば、ユーザ役割は、分散システムまたはウェブ・ブラウザ上のメニュー項目をグレーにする(使用不可状態にする)ためユーザ役割を使用することができる。ユーザ役割は、指定された役割におけるユーザの責任に関してアクセス規則が特に作成されることを可能にするので、動的な役割切り替えが可能で、権限の過大付与または過小付与に関連する問題を除去することができる。ユーザが複数アプリケーションの間で許可されたことだけを一貫して実行することができることを保証するため本発明を使用することができる。権限が複数のレイヤではなく単一のレイヤによって実施される場合、権限の矛盾した権限の内容がユーザに与えられる可能性は少ない。更に、矛盾した権限の適用を除去することによって、整合性のある権限許可または共用オブジェクトへのアクセスの否定を通してアプリケーションの統合がサポートされる。

【００１４】本発明の第１の側面において、オブジェクトおよびドメインの概念が存在しない場合のユーザ役割アクセス制御を実施するため、ウェブに基づくモデルが使用される。本発明の第２の側面において、ユーザ役割アクセス制御を実施するため、ユーザ役割がオブジェクトおよびドメイン概念と相互に関係づけられる。上記両方の側面において、ユーザ役割アクセス制御は、グラフィカル・ユーザ・インタフェース(すなわちＧＵＩ)フィルタリングおよび選択されたアクションの実行時アクセス検査という２つの形態で使用される。この両者は、どのＵＲＬまたはメニュー項目が例えば表示されアクセス権を与えられるかを決定するユーザ役割のアクセス権によって定義される。

【００１５】過去におけるようなアクセス権をオブジェクトまたはユーザに関連づける方法に比較してユーザ役割アクセス制御は一層柔軟である。アクセス権をオブジェクトに関連づけることは、オブジェクトへのアクセスが新しいユーザに与えられる必要がある場合、構成の修正を困難にする。多数のオブジェクトが存在し、(一人が同僚の役を引き受ける時のように)新しい権利が一時的のものにすぎないような場合特に困難である。また、(動的ネットワーク環境においては頻繁に起きることであるが)新しいオブジェクトが追加される場合や、(病気や休暇でだれかの代役を務めなければならないような場合のように)果たさなければならない異なる役割に基づいてユーザがアクセス権を変更する必要がある場合、権限のユーザへの関連づけはその維持が困難である。

【００１６】ユーザ役割アクセス制御は、最高の柔軟性を提供し、構成の複雑さおよびオーバーヘッドを最小限にとどめる。例えば、上記第２の側面において、(例えば新しいオブジェクトの追加によって)ドメインの定義を変更することは、そのドメインに関するアクセス規則を含むユーザ役割のすべてに影響を及ぼす。(例えば新しいアプリケーションの導入のため新しい操作の追加によって)操作セットの定義を変更することは、その操作セットに関するアクセス規則を含むユーザ役割のすべてに影響を及ぼす。(例えばアクセス規則の追加または削除によって)ユーザ役割の定義を変更することは、そのユーザ役割に割り当てられたすべてのユーザに関するアクセス権を変更する。影響を受けるユーザ各々についてアクセス権を更新する必要はない。更に、ユーザ役割アクセス制御は、大きな、明確に定義されたグループにまとめられたアクセス権をユーザに付与または否定することを容易にする。例えば、種々のオブジェクトに関する種々の操作に関して多数の個別的アクセス権を割り当てる代わりに、"ネットワーク操作員"ユーザ役割をユーザに割り当てることによって、"ネットワーク操作員"権限をそのユーザに与えることができる。

【００１７】

【発明の実施の形態】本発明は、ユーザ役割を通してユーザ・アクセス制御を実施する方法および装置である。本明細書において、"ユーザ"は、人、または、特定の要求に関してユーザ役割を継承するサーバのいずれかを意味する。ユーザは、ユーザ認証のために使用されるユニークなＩＤによって識別される。ユーザ役割は、ユーザの機能が何であるかを決定するアクションのセットである。ユーザ役割は、セキュリティ管理者、ネットワーク管理者またはシステム管理者のような操作上の責任分野を定義することができる。

【００１８】ユーザ役割アクセス制御は、(図２に示されているような)１つまたは複数のコンピュータ読取可能記憶媒体２３２で実施されることができ、１つまたは複数のユーザ役割２００、２０２、２０４を作成するプログラム・コードが上記１つまたは複数のコンピュータ読取可能記憶媒体２３２上に存在する。１つまたは複数のユーザ役割２００、２０２、２０４の各々は、ユーザ・セットに関連するアクセス権セット２２６、２２８、２３０を含み、アクセス権２２６、２２８、２３０の各々は、ユーザ・セットの中のユーザによってコンピュータ・オブジェクト・セットに関して実行されるアクション２０６、２０８、２１０、２１２、２１４、２１６、２１８、２２０、２２２を記述する。

【００１９】ユーザ役割アクセス制御は、また、ユーザ・アクセスを制御する方法の形態で実施されることもできる。この方法は、コンピュータ・セッションの間にアクションを実行すべき要求をユーザから受け取り、その要求に応答して、上記ユーザが当該セッションに関してそのメンバである１つまたは複数のユーザ役割２００、２０２、２０４を取り出すステップを含む。この場合、１つまたは複数のユーザ役割２００、２０２、２０４の各々は、ユーザが当該セッションに関して有する１つまたは複数のアクセス権２２６、２２８、２３０を決定し、その１つまたは複数のアクセス権２２６、２２８、２３は、当該セッションの間に当該ユーザが実行することができる１つまたは複数のアクション２０６、２０８、２１０、２１２、２１４、２１６、２１８、２２０、２２２を決定する。該方法は、更に、当該ユーザがそのメンバである１つまたは複数のユーザ役割２００、２０２、２０４が要求されたアクションを許可するアクセス権２２６、２２８、２３０を含むか否かを判断することによって当該ユーザが要求されたアクションを実行する権限を与えられているか否かを判断するステップを含む。

【００２０】図１は、ユーザ・アクセス制御のドメイン・モデルを図示している。このモデルにおいては、オブジェクト・ドメイン１０６、１０８、１１０に対するユーザのアクセス権は、アクション・セット１００、１０２、１０４の１つまたは複数ペアリングによって決定される。このモデルにおいては、要求されたオブジェクトを含むオブジェクト・ドメインに対する要求されたアクションを含む操作セットのペアリングを定義するアクセス権をユーザが割り当てられている場合に、ユーザはそのオブジェクトに関するアクションに対する権限を与えられる。

【００２１】図２は、本発明のユーザ役割アクセス制御の実施形態を示している。本発明のユーザ役割アクセス制御においては、ユーザが、１つまたは複数のアクション２０６、２０８、２１０、２１２、２１４、２１６、２１８、２２０、２２２に対する１つまたは複数のアクセス権２２６、２２８、２３０を含む１つまたは複数のユーザ役割２００、２０２、２０４のメンバであれば、そのユーザの特定の要求に権限が付与される。ただし、この場合、少なくとも１つのアクセス権が要求されたアクションを含む。言い換えると、要求されたアクションに対する少なくとも１つのアクセス権を含むユーザ役割にユーザが関連づけられていれば、そのアクションは許可される。

【００２２】第１の好ましい実施形態

本発明の第１の好ましい実施形態において、ユーザのアクション起動を許可または否定するユーザ役割アクセ制御がウェブ型環境において実施される。この場合、ＵＲＬが特定のアクションに対するユーザのアクセス権を決定する。ウェブ型ユーザ役割モデルに関するこの実施形態は、ウェブ・ブラウザおよびウェブ・サーバとは独立したメカニズムである。ウェブ型ユーザ役割アクセス制御は、ランチャと呼ばれる１つのウェブ・アプリケーションの形態で実施され、ＵＲＬを通してユーザにアクションの実行を可能にさせる。

【００２３】本発明の１つの側面において、活動的ユーザ役割に関して定義されるアクションに対してＧＵＩをフィルタリングするため、ユーザ役割アクセス制御がランチャ・ウィンドウの表示に適用される。ユーザは、表示されるいかなるＵＲＬをも選択することができるように認められる。別の側面においては、ユーザ役割アクセス制御が実行時アクセス検査のため使用される。この場合、アクションに関するＵＲＬがユーザのセッションの１つまたは複数のユーザ役割に関して定義されていれば、そのユーザはそのアクションを実行する権限を与えられる。

【００２４】ユーザ・ログインおよびセッション初期設定

１つまたは複数のユーザ役割によって定義されるアクセス権を得るように認証され、権限を取得するためには、ユーザはウェブ・セッションを開始しなければならない。セッションは、特定の表示上の特定のユーザに関連するアプリケーションのグループである。(例えばウェブ・セッションのための最上位ＧＵＩのような)ランチャ・ウィンドウから直接または間接始動されるアプリケーションがセッションの一部であり、情報を共有することができる。所与のウェブ・ブラウザ・プロセスに関してログインが行われていない場合、あるいは、ウェブ・セッションが活動の構成可能期限をすぎた場合、新しいウェブ・セッションを開始することができる。ウェブ・セッションの状態の一部は、ユーザのＩＤ(ユーザ名)およびユーザが実行の資格を持つものを定義するユーザ役割のリストである。一旦ウェブ・セッションが作成されたならば、ユーザは、そのセッションに関するユーザ役割によって許可されるＵＲＬに自由にアクセスすることができる。このようなユーザ役割は、ユーザの活動的ユ

ーザ役割と呼ばれる。このような活動的ユーザ役割は、どのＵＲＬが表示されるかという観点からどのような種類のユーザ・インタフェースをユーザが見るかを決定する。このユーザ役割は、また、ユーザが起動することができる実行時ＵＲＬを決定する。この実施形態において、該ユーザに対するユーザ役割のすべてが活動的である場合、ユーザは、該ユーザがそのメンバであるユーザ役割のすべてを与えられる。本発明は、また、ユーザの権限を制限または拡張するため、どのユーザ役割が活動的でありどのユーザ役割が非活動的であるかをユーザが選択することを可能にする概念を含む。

【００２５】図３において、ユーザは、ランチャに関するＵＲＬをウェブ・ブラウザ３００に入力することによってセッションを開始する。例えば
"http://hostname/OvCgi/ovlaunch.exe"
のようなランチャに関するＵＲＬが起動される。このＵＲＬは、セッションが開始する時アクセス権限に応じて実行することができるＵＲＬ引数を受け入れることができる(ＵＲＬがovlaunchに指定されない場合、ログインの後デフォルト・ランチャ・ウィンドウが表示され、そこでユーザはランチャを通してまたはウェブ・ブラウザを通してＵＲＬを起動することができる)。ランチャに関するＵＲＬ要求は、ワールド・ワイド・ウェブにおいて情報にアクセスするために使用されるクライアント／サーバ・プロトコルを通してウェブ・サーバ３０２へ転送される(図３のステップ３０４)。このプロトコルは、ハイパーテキスト・トランスファ・プロトコル(すなわちＨＴＴＰ)と呼ばれる。ウェブ・サーバは、ＨＴＭＬログイン画面を提示しユーザ名およびパスワードの入力をユーザに促すＣＧＩ(すなわちCommon Gateway Interface)プログラムのovlaunchを起動する(ステップ３０８)。画面がウェブ・サーバに送り戻される(ステップ３３８)。

【００２６】ウェブ・サーバは、画面を読み取り、ovlogin３１２を起動し(ステップ３１０)、このブラウザに関するセッションが存在するか否かを判断するためユーザによって入力されたユーザ名およびパスワードを渡す。この判断は、OvWebSessionと呼ばれるブラウザ・クッキーが設定されているか否か検査することによって行われる(詳細は後述)。このクッキーが設定されていれば、ユーザが既にログインされていることを標示するメッセージが返され、ovloginはクッキーからセッション番号を取得して、ovsessionmgrを用いてセッションを検証するためそのセッション番号を使用する。クッキーが設定されてなければ、ovloginは新しいセッションを要求する。両方のケースにおいて、ovloginは、セッション情報に関してサーバの役割を果たすovsessionmgr３１６へ接続する(ステップ３１４)ことを試みる。ovsessionmgrへ接続されると、ovloginは、クッキー・セッション番号を渡す(ステップ３１８)ことによって既存のセッションを検証するか、または、現在時環境によって定まる遠隔ＩＰアドレスと共にユーザ名およびパスワードをovsessionmgrに渡す(ステップ３１８)ことによって新しいセッションを要求する。

【００２７】ユーザ名およびパスワードが有効でないとすれば、ovsessionmgrはこの情報をovloginに中継し(ステップ３２８)、ovloginはそれをウェブ・サーバに送る(ステップ３４０)。loginページは、ウェブ・ブラウザによって受け取られる(ステップ３０４)エラー標示によって置き換えられ、次に表示される。ユーザ名およびパスワードが有効であれば、現行セッションは検証されつつあり、ovsessionmgrは、ユーザが既にログオン済みであることを標示するメッセージをovloginに送り返す(ステップ３２８)。次に、ovloginは、そのメッセージをウェブ・サーバ経由(ステップ３４０)でウェブ・ブラウザに送り戻す(ステップ３０４)。ユーザ名およびパスワードが有効で、新しいセッションが要求されていれば、ovsessionmgrは、ログインが成功したことを標示するメッセージをovloginに送り(ステップ３２８)、次にovloginがそのメッセージをウェブ・サーバ経由(ステップ３４０)でウェブ・ブラウザに送り戻す(ステップ３０４)。

【００２８】ログインが成功していれば、ovsessionmgrは、グローバル・セッション情報を維持するセッション構成ファイル３２０を読む。Ovsessionmgrは以下のフィールドに関する値を受け取る(ステップ３２１)：

・UserLogin：ユーザ認証の有無にかかわらずセッションが作成されることを可能にする。値はon/offであり、デフォルト値はoffである。

・LoginLogging：成功および失敗したログイン試行を記録する。値はon/offであり、デフォルト値はoffである。

・AccessLogging：アクセスが成功したＵＲＬを記録する。値はon/offであり、デフォルト値はoffである。

・セッション・タイムアウト：セッションがタイムアウトとなる時間量を指定する。値は０より大きな整数であり、デでフォルト値は９である(単位は時間)。

【００２９】これらの値のいずれかが最後のセッションから変わったならば、ovsessionmgrはその内部構成値を更新する。UserLoginがoffであれば、それ以上の認証を必要とすることなくセッションが作成される。UserLoginがonであれば、ovsessionmgrは、管理者によって設置されhtpasswdプログラム３２４によって記憶された(ステップ３２３)暗号化されたユーザ・パスワードのレポジトリであるユーザ・パスワード・ファイル３２２を読み取ることによってユーザを認証する。ユーザ・パスワード・ファイルは、"banana:FXDFRAxjRkuFA"という形式を持つ。この第１の値はユーザ名であり第２の値は暗号化されたパスワードである。

【００３０】ユーザ・パスワード・ファイルによって返された(ステップ３２５)パスワードが、ユーザが入力し

たパスワードと一致すれば、ユーザは認証され、セッションが作成され、以下のセッション情報がovsessionmgrに関連して記憶される。

・ユーザ名

・パスワード：パスワードは、(例えば各文字に対する論理的ＸＯＲを行うことによって)プロセス・メモリの読み取りによって明示的テキストで見ることができないような形式で記憶される。

・ユーザ役割：ovsessionmgrは、どのユーザがどのユーザ役割に属しているかを指定するhtpgroupファイル３２６を読むことによってこの情報を取得する。ただし、この場合、各ユーザは１つまたは複数のユーザ役割に属していることができる。ユーザ役割ファイルは、

my-users：pumpkin peanuts almonds walnuts

という形式を持つ。この形式において、最初の値はユーザ役割であり、後続の値は、そのユーザ・グループに属するユーザである。ユーザ役割ファイル３２７は、現在ユーザがそのメンバであるユーザ役割をovsessionmgrに送り返す(ステップ３２７)。その結果、ユーザは、そのユーザ役割のすべてに関するアクセス権のすべてを与えられる。しかしながら、本発明は、また、ユーザがユーザ役割にログインし、複数の役割の間で切り替えを行うことを可能にする。

遠隔ＩＰアドレス：これは、セッション番号に基づいて呼び出し元の評価を更に実施するため使用される。

【００３１】ウェブ・セッションに属している(すなわちウェブ・セッションの文脈の範囲内で開始された)ウェブ・アプリケーションは、現行ユーザに関するセキュリティおよびアクセス制御情報に対するアクセス権を有する。(ウェブ・アプリケーションは、例えば、JavaアプレットおよびＣＧＩプログラムである。) この情報は、セッション・プロパティを通してウェブ・アプリケーションにとって利用可能となる。このセッション・プロパティは、アプリケーション・プログラミング・インタフェース(すなわちＡＰＩ)を介してウェブ・セッションに属しているどのようなアプリケーションもアクセスすることができる。図４は、セッション特性にアクセスするためのJavaクラスのサンプル定義であり、図５は、セッション特性にアクセスするためのＣ言語ＡＰＩのサンプル定義である。

【００３２】図４において、セッションが初期設定され(４００)、現在時セッションに関するセッションＩＤがランチャから取得される(４０６)。セッションが有効でないとすれば(すなわちユーザが認証されていなければ)、Javaアプレットから出るか、他の適切な措置が取られる(４０２)。一旦セッションが始まると、ログインが使用可能とされていれば(４０４)、Javaアプレットはユーザのログイン名を取得することができる(４０８)。次に、Javaアプレットは、ログインが使用不可にされてない限りそのユーザに関するユーザ役割のリストを取り出すことができる(４１０)。このJavaアプレットは、また、実行時アクセス検査を実行することができる(４１２)。この検査は、ＧＵＩを通して視認できるものより一層きめこまかいアクセス制御を可能にする。

【００３３】図５において、ＣＧＩプログラムは、確保した変数が真であればユーザが既にログインしているか否かを検査する(５００)。ユーザがまだログオンしていなければ、確保した引数を真にセットしてOVwwwInitを呼出し、ログイン・ページをユーザに提示する。ユーザがログインに成功した後、ＣＧＩプログラムは再び呼び出される。図４のJavaアプレットと同様に、セッションが有効でなければ、このＣＧＩプログラムから出るか、または、他の適切な措置が取られる(５０２)。セッションが有効であれば、セッションＩＤが取得される(５０４)。ログインが使用可能であれば(４０６)、ＣＧＩはユーザ名を取り出す(５０８)。ＣＧＩプログラムは、また、ユーザの役割を取り出し(５１０)、ユーザが特定の役割に属しているか検査する(５１２)。このＣＧＩは、また、実行時アクセス検査を実施することができる(５１４)。

【００３４】図３を再び参照すれば、セッション・プロパティLoginLoggingが使用可能とされていれば、ログ・メッセージがログイン・ログ・ファイル３３２に書き込まれる(３３０)。新しいセッションを作成する時、ovsessionmgrは、ランダムに生成されたセッション番号をovloginに返し(３２８)、ユーザがログインしたという証明の役目を果たすブラウザ・クッキーOvWebSessionを作成する。ユーザがブラウザから出る時、あるいは、非活動の構成期間の後セッションが時間切れとなった時、クッキーは無効となりセッションは終了する。一旦ovsessionmgrがセッションの後の評価のためブラウザ・クッキーOvWebSessionにセッション番号を記憶すると(このクッキーは現行セッションを検査するためovloginによって使用される)、ユーザは、ＵＲＬを実行することによってアクションを実行することができる。

【００３５】ランチャ・ウィンドウからＵＲＬを選択するか、ユーザのログインと共に自動的に実行されるようにランチャにＵＲＬ引数を与えるか、または、ウェブ・ブラウザからＵＲＬを直接起動することによって、ＵＲＬは実行されることができる。ＵＲＬを選択するという第１の方法におけるアクセス権限は、ランチャ初期設定時に定義される。この場合、ユーザの役割によって認可されるＵＲＬに対してＧＵＩをフィルタリングすることによってユーザ役割アクセス制御が実施される。後者の２つの方法におけるアクセス権限はＵＲＬ実行時に定義される。この場合、ユーザ役割アクセス制御は実行時アクセス検査によって実施される。

【００３６】ランチャ初期設定およびＧＵＩフィルタリング

(ログイン・プロシージャの後)ovlaunchＣＧＩプログラ

ムを実行するデフォルト動作はランチャ・ウィンドウを立ち上げることである。(デフォルト動作は、いかなるＵＲＬ引数をも必要とすることなくＣＧＩプログラムovlaunchを要求することを指す。) ランチャ・ウィンドウは、現在時セッションに関してユーザ役割によって許容される機能性すべてをユーザが利用することができるようにするＧＵＩである。ランチャ・ウィンドウは、ユーザ役割構成によってユーザに許容される操作を提示する多くの可能な方法の１つである。一般的概念は、ユーザが権限を与えられる操作だけをＧＵＩがユーザに提出するようにフィルタリングを行うことである。

【００３７】ランチャ・ウィンドウは、ユーザ役割に基づく管理機能性(すなわちアクション)の表示を提供し、ＵＲＬを介しその機能性を起動するJavaアプレットである。ランチャ・ウィンドウは、その内容がランチャ登録ファイルから取得される２つの部分、すなわち、管理機能性および活動的ヘルプに分割される。管理機能性領域は、管理操作のカテゴリを標示するタブ・セットを含む。各カテゴリは小さいアイコンによって標示される。カテゴリのリストは、例えば、タスク、ツール、オブジェクト・ビュー、管理領域およびヘルプを含む。各カテゴリの範囲内に、関連した操作の階層的な(ツリー・リスト)表現が存在する。これらのツリー・リストの内容は、ユーザのユーザ役割(1)およびランチャ・セッションの局地性(2)によって決定される。ユーザが関連づけられるユーザ役割がランチャ・ウィンドウに現れる操作を決定するが、ユーザが役割のサブセットを選択することもできる。

【００３８】ユーザは、コンテナ・ノードを開いたり閉じたりする標準メカニズムを使用してツリー・リストを逐次調べる。コンテナ・ノードを１回クリックすればノードが広がる。葉ノードを１回クリックすれば、ＵＲＬを実行するためのものである関連アクションが起動される。ランチャ・カテゴリが空であれば(すなわちその下にリスト項目がなければ)、カテゴリはユーザに提示されない。ランチャ・コンテナ・エントリが空であれば、コンテナ・エントリは表示されない。(これは空のメニュ・カスケードに似ている。コンテナの下にただ１つの項目が存在すればメニューの崩壊はない。) 活動的ヘルプ領域は、カーソルが置かれる操作の機能を説明する短いヘルプ・メッセージを表示する。

【００３９】ランチャ・ウィンドウに表示されるアクションは、セッションに関する活動的ユーザ役割に依存する。ユーザ役割は、通常は開発者によって提供されるが管理者によって修正または補足されることができるランチャ登録ファイルを介して構成される。ランチャ登録ファイルは以下の項目を含む：

・表示文字列、バージョン、著作権および説明を含むアプリケーション情報。このブロックはランチャによって表示されないが、ランチャ登録ファイルにおける有用な情報を提供するためにだけこの項目は使用される。

・タブ・ブロックは、オプションのアイコン・ファイル名がリスト項目エントリを含むタブ上に表示されることを可能にする。このブロックは、また、ユーザがタブを選択する時に提示されるオプションの活動的ヘルプ・テキストを含む。

・リスト・ブロックはリスト項目エントリを含む。リスト項目のコンポーネントは、優先順位値、リスト項目名、アイコン、活動的ヘルプおよび機能を含む。２つの可能な機能は、アクション機能およびリスト機能である。アクション機能は、アクション定義を持つアクション・ブロックを指し示す終端リスト項目を示す。リスト機能は、階層的ツリーの定義を可能にするため、リスト・ブロックを指し示すコンポーネント・リスト項目を示す。

・アクション・ブロックは次のようないくつかのステートメントを含む。－アクションを起動するＵＲＬを含むＵＲＬステートメント。－このアクションに対するアクセス権を持つユーザー役割のリストであるアクセス・ステートメント。アクセス・ステートメントが存在しなければ、すべての有効なユーザがアクションにアクセスすることができる。ユーザ役割アクセス制御の要点はこのステートメントで定義される。－ＵＲＬがそこにロードされるウィンドウの特性を指定するWebWindowステートメント。

【００４０】図６は、ランチャ登録ファイルのサンプルであり、これは、アプリケーション情報６００、リスト項目エントリおよび２つの活動的アイコンを定義するタブ・ブロック６０２、アクション・ブロックを指し示すリスト・エントリを含むリスト・ブロック６０４、および、各々がそのアクションに関して起動されるべきＵＲＬおよびそのアクションを実行する権限を与えられるユーザ役割を含む２つのアクション・ブロック６０６(ａ)および６０６０(ｂ)を含む。

【００４１】ランチャ初期設定の間にＧＵＩフィルタリングによって実施されるウェブ・ブラウザにおけるユーザ役割アクセス制御が図７に示されている。ユーザは、ウェブ・サーバ３０２に要求を送る(３０４)ウェブ・ブラウザ３００を経由してovlaunchＵＲＬを起動する。ウェブ・サーバ３０２はovlaunchreg７０２を起動して(７００)、ランチャ登録ファイル７０６から登録情報を取り出す(７０４)。Ovlaunchregは、次に、ovsessionmgr３１６に関連して記憶されている情報から７０８のセッション情報(すなわちユーザおよびユーザ役割)を取り出す(７０８)。Ovlaunchregは、構文解析ライブラリを呼び出して、登録情報を解析し、ユーザが認可されているユーザ役割に基づいてそれをフィルタリングする。その関連アクションに関してユーザが認可されていれば、エントリは、ウェブ・サーバ３０２、ウェブ・ブラウザ３００およびランチャ・ウィンドウ７１４にそれぞれ返さ

れる(７１０、３０４、７１２)。いくつかのツリー・リストが、ランチャ・ウィンドウにおいて、各々タブを付けられた枠の上に構築される。結果として、ユーザは、認可されているものだけを見るので、ランチャ・ウィンドウにおいて使用可能となっているすべてのＵＲＬに対するアクセス権を持つ。

【００４２】アクション呼び出しおよび実行時アクセス検査

ユーザに関する関連操作のＧＵＩフィルタリングに加えて、ユーザが実際にアクションを起動する時の実行時アクセス検査によってアクセス制御を実施することができる。ユーザは、以下の最初２つの方法によって示されるようにＣＧＩプログラムovlaunchの使用を通して、あるいは、最後の方法によって示されるようにovlaunchを使用せずにアクションを起動することができる。

【００４３】ユーザがアクションを起動することができる１つの方法は、ＵＲＬおよびその関連したアクションに対応するランチャ・ウィンドウにおける項目をクリックすることによるものである。ＧＵＩフィルタリングが実施されていなくても、あるいは付加的セキュリティ・レイヤとしてさえ、実行時アクセス検査をこのシナリオで実施することができる。アクション起動の結果、引数として要求されたアクションのＵＲＬを使用するovlaunchＣＧＩプログラムが起動される。ovlaunchＣＧＩプログラムは、Ｃ言語ウェブ・セッションＡＰＩを使用して、有効なセッションがあることを検証するためovsessionmgrプログラムと対話する。ovlaunchＣＧＩプログラムは、また、ＡＰＩ呼び出しを使用して、そのセッションに関する活動的ユーザ役割に基づいて特定のアクションを実行することをユーザが許可されていることを検証する。

【００４４】ユーザがアクションを起動する別の１つの方法は、ウェブ・ブラウザを使用してＵＲＬ引数でovlaunchに関するＵＲＬを起動するものである(例えばhttp://OvCgi/ovlaunch.exe?URL=http://some/path)。そこで、Ovlaunchは、ＵＲＬによって指定された要求アクションを開始するが、そのアクションはセッションに関するユーザのアクセス権に依存する。これは、ブックマークとして保存されるアクションに対するアクセス権を得るため役立つ。この動作は、また、ランチャ・ウィンドウの代わりに"代替コンソール"を開始するため使用することができる。

【００４５】最後に、ユーザは、ウェブ・ブラウザから直接その関連ＵＲＬを要求することによって(ovlaunchＣＧＩプログラムの介在なしに)アクションを起動することができる。これは、制限された機能性にアクセスする潜在的"後方ドア"メカニズムを構成するので、付加的セキュリティ・メカニズムが必要である。第１に、ウェブ・ブラウザから見えるように構成されてない特別な"保護された"ディレクトリにＨＴＭＬドキュメントが置かれる。そこで、ovlaunchＣＧＩプログラムに対するＵＲＬ引数としてそれらドキュメントを要求することによってのみこれらＨＴＭＬドキュメントへのアクセスが可能となる。ovlaunchＣＧＩプログラムは、ＨＴＭＬドキュメントに関する"保護トされた"ディレクトリを検査して、ユーザが適切なアクセス権を持つとすればそれを返す。"保護された"ディレクトリにおけるＨＴＭＬドキュメントは、それらを要求することによって直接取り出さすことはできない。第２に、ウェブ・アプリケーション(本実施形態におけるＣＧＩプログラムおよびJavaアプレット)が、ウェブ・セッションＡＰＩを使用して実行時アクセス検査を行うことができる。セッションが存在すること(すなわちユーザが既にログインして認証されている)、および、要求されているアクションが現在時ユーザに関するユーザ役割の中にあることをこれらプログラムは確認することができる。ユーザがウェブ・アプリケーションによって表されるアクションに関する許可を取得していないとすれば、プログラムから出るか、他の適切なアクションをとることができる。

【００４６】ユーザはovlaunchＣＧＩプログラムを潜在的にバイパスすることができるけれども、このプログラムはいくつかの理由から必要とされる。第１に、ovlaunchＣＧＩプログラムは、ＵＲＬを実行するための特別仕様のウェブ・ブラウザ・ウィンドウの作成のような、セキュリティに関連しない他の役割を提供する。第２に、ovlaunchは、"保護された"ディレクトリにおけるＨＴＭＬドキュメントに対する安全なアクセスを提供する。第３に、ovlaunchは、ウェブ・セッションＡＰＩを使用するように修正されていないウェブ・アプリケーションに関する一定レベルのアクセス検査を提供する。ユーザがアクションを起動するためのＵＲＬを知らないとすれば、これは特に真である。これは、既存のウェブ・アプリケーションの統合を一層容易にする。しかしながら、一層高いレベルのセキュリティに関する限り、ウェブ・アプリケーションがユーザ役割セキュリティに関連するウェブ・セッションＡＰＩを使用することが必要である。

【００４７】図８は、ＣＧＩプログラムovlaunchを使用してＵＲＬを介してユーザがアクションを起動する際実行時アクセス検査によって実施されるウェブ・ブラウザにおけるユーザ役割アクセス制御を示す。フィルタリングされてないメニュー項目をクリックすることによって、あるいは、ovlaunchＵＲＬにＵＲＬ引数を直接提供することによってウェブ・ブラウザ３００を経由して、ユーザは、ランチャ・ウィンドウ７１４を介してアクションを起動することができる。両方の要求はウェブ・サーバ３０２に送られ(３０４，８１６)、結果としてＣＧＩプログラムovlaunchが実行される(３３８)。セッションが存在することを検証した後、ovlaunchは、ovsessionmgrに照会する(８０２)ことによって、要求されたアク

ションがユーザの活動的ユーザ役割のうちの１つによって許可されているか否か検証することができる。Ovsessionmgrは、セッション初期設定時にovlaunchreg７０２から取得した(７０８)ユーザ役割情報をキャッシュ記憶した後、その結果をovlaunch３０８に返す(８０２)。ユーザがアクション実行の権限を与えられていれば、ovlaunchregはウェブ・ブラウザへＨＴＭＬページを返し(３３８、３０４)、その結果、起動されたＵＲＬによって標示されるＣＧＩプログラム８１０の実行を要求する(８１２)要求が自動的に出される(３０４)。

【００４８】図９は、ＣＧＩプログラムovlaunchを使用せずに、ＵＲＬを介してユーザがアクションを起動する際実行時アクセス検査によって実施されるウェブ・ブラウザにおけるユーザ役割アクセス制御を示す。ユーザが、ＣＧＩプログラムovlaunchを使用せずに、ウェブ・ブラウザ３００を通してＵＲＬに直接入力すると、ウェブ・サーバ３０２は、起動されたＵＲＬによって標示されるＣＧＩプログラム８１０を実行する(８１２)。起動されたＵＲＬのＣＧＩプログラムは、ウェブ・セッションＣ言語ＡＰＩを使用して、ovsessionmgr３１６と通信して、セッションが存在すること、および、要求されているアクションが現在時ユーザに関するユーザ役割によって許可されていることを検証する。(前述の場合と同様に、ovsessionmgrは、セッション初期設定時にovlaunchreg７０２からユーザ役割情報を取得して、キャッシュ記憶する。) セッションが存在し、ユーザがアクションに関して認可されていれば、ＣＧＩプログラムは、設計されたアクション(例えばＵＲＬ実行)の実行を続行する。

【００４９】第２の好ましい実施形態

本発明の第２の好ましい実施形態において、ユーザのアクション起動を認可または否定するユーザ役割アクセス制御が任意のネットワーク環境において実行される。この場合、アクセス規則がアクションに対するアクセス権をユーザに与える。アクセス規則は、特定の操作セットと特定のドメインの間の関係である。操作セットはその英語表現Operation Setを短縮して以下op-setと呼称する場合がある。操作セットは、オブジェクトに関して取られることができるアクションのセットである。大部分の操作はオブジェクトの特定クラスに関するものであるが、一部の操作はオブジェクトに特定しない一般的操作である。操作の例には、システムのリブート、ＡＴＭスイッチの追加、ルートの検出およびレコードの追加が含まれる。操作セットは、典型的には、プリンタの管理に必要とされるすべてのアクションのような論理的に関連するアクションのグループである。ドメインは、関心のある分野および責任分野に細分化することによって管理環境のサイズおよび複雑さを減らすため使用されるオブジェクトのセットである。ドメインは、管理される目標オブジェクトを定義し、ドメイン内のすべてのオブジェクトに何らかの共通のアクセス制御ポリシーを適用する目的からグループ化される。ドメインは、地理、ネットワーク・トポロジ、機能、組織のような種々の目的のため定義される。ドメインは、(例えば"すべてのルーター"あるいは"WindowsNT4.0を実行するすべてのＰＣ"のような)規則セットの観点から定義されるか、または明示的に定義されるメンバで定義される。

【００５０】典型的には、管理されるオブジェクトはユーザ役割定義に暗示されている。例えば、１つのユーザ役割が米国東部に関するネットワーク管理者であり、別のユーザ役割が米国西部に関するネットワーク管理者である。要求されたオブジェクトに対して要求された操作が実行されることを許可するアクセス規則を含むユーザ役割をユーザが持つとすれば、そのユーザはオブジェクトに対する操作の実行の権限を与えられている。例えば、ＡがＰのメンバでありＸがＱのメンバであるように、操作セットＰとドメインＱのペアとして定義されるアクセス規則を含むユーザ役割をあるユーザが与えられたとすれば、そのユーザはオブジェクトＸに対する操作Ａを実行する権限を与えられる。本実施形態は、ユーザ・アクセス権限の上に更にユーザ・アクセスに対する制約を付加することを可能にする。例えば、時間制約が午前８時０分と午後５時０分の間にユーザがログオンするように制限し、位置制約が、一定の物理的に安全保護されたシステムへログオンする時にユーザがアクセス権を取得するように制限することができる。

【００５１】本実施形態におけるユーザ役割アクセス制御は、ヒューレット・パッカード社のOpenViewアプリケーションにおいて実施されることができる。ユーザがOpenViewアプリケーションへログオンする時、ユーザは、この実施形態における１つまたは複数のユーザ役割を引き受ける。ユーザは、ユーザが許可を与えられたユーザ役割のすべてである付与されたユーザ役割を持つ。ログインする時、ユーザは、付与されたユーザ役割のサブセットを請求する。これは、請求されたアクセス権または活動的ユーザ役割と呼ばれる。活動的ユーザ役割が、どのメニューが表示されるか、どのオブジェクトが見えるかという観点からどのような種類のユーザ・インタフェースをユーザが見るかを決定する。活動的ユーザ役割に関する操作セットにおける操作のすべてがメニューおよびボタンのようなＧＵＩ制御機構を介して使用可能とされる。活動的ユーザ役割の操作セットに現れない操作はＧＵＩ制御機構に現れない。ユーザが新しい役割を活動的ユーザ役割のリストに追加すれば、それら操作に関するＧＵＩ制御機構がユーザ・インタフェースに現れる。同様に、活動的ユーザ役割に関するアクセス規則に含まれるドメインの各々におけるオブジェクトのすべてがユーザ・インタフェースを通して表示される。これらのドメインにないオブジェクトはユーザ・インタフェースを通して表示されない。ユーザの活動的ユーザ役割は、ま

た、ユーザがどの実行時操作を実行することができるかを決定する。本実施形態において、(同僚が病気の時のように)ユーザがたまに一定の役割を必要する時に役立つ特権の追加または削除を行うため、ユーザは役割を動的に切り換えることができる。

【００５２】ユーザ・インタフェースにおける操作およびオブジェクトの存在は、操作およびオブジェクトが異なるアクセス規則に属することができるので、ユーザがそのオブジェクトに関するその操作を実行することができることを意味しない。例えば、オブジェクトＸに対する操作Ａおよびオブジェクト Ｙに対する操作Ｂをユーザが実行することを許容するアクセス規則が存在しても、これは、ユーザがオブジェクトＹに対する操作Ａを実行することができることを意味しない。このような理由から、また、セキュリティ管理者がオペレータのアクセス権を変更するかもしれないという事実のため、本発明は、実行時アクセス検査を含む。ユーザがオブジェクトＸに対するアクションＡの実行を要求する時点で、Ａが活動的ユーザ役割に関する操作セットのメンバであり、Ｘがそのアクセス規則に関するドメインのメンバであるというアクセス規則(すなわちop-set/ドメイン・ペア)が活動的ユーザ役割の１つに存在するか否か検査される。両方の条件が存在すればアクションが取られる。

【００５３】図１０は、本発明の第２の好ましい実施形態のセキュリティ・モデルのサンプルを示す。このモデルは、ユーザ役割１０００、操作セット１００２およびドメイン１００４を定義する。この例では、ユーザJoe SmithおよびSue Edwards１０１４がユーザ役割NFS_Administrator１００６のメンバである。彼らは、１つまたは複数のオブジェクト・ドメイン１０１におけるオブジェクト・セットに対する１つまたは複数の操作セット１００８における操作セットを実行することのできるアクセス規則１０２４、１０２６、１０２８、１０３０を有する。op-setは、例えば、op-setNFS_Admin１０１６がＮＦＳサーバ読み取り、ＮＦサーバ構成、ノード読取りおよび可視性という操作セットを含むように定義されている。ドメインもまた、例えばdoc_serv1、ノード東部、doc_serv1:/、doc_serv1：/ProjDocsを含むように、定義されている。従って、例えば、Joe Smithがユーザ役割NFS_Administratorにログインして、ドメイン東部に対して(NFS_Admin操作セットに定義されている)ＮＦサーバ読み取り操作を実行しようとすれば、アクセス規則定義 {NFS_Admin,東部} が彼にアクセス権を与える。一方、彼がドメイン南部に対して同じことをしようとすると、{NFS_Admin,南部} を定義するアクセス規則が存在しないため、アクセスは否定される。

【００５４】図１１は、本発明の第２の好ましい実施形態におけるユーザ役割アクセス制御を更に例示している。ここでは、ユーザ１１００、１１０２、１１０４が１つまたは複数のユーザ役割１１１４、１１６に割り当てられている(１１０６、１１０８、１１１０、１１１２)。各ユーザ役割は、１つまたは複数のアクセス規則１１１８、１１２０、１１２２、１１２４すなわち操作セット１１２６、１１２８のドメイン１１３０、１１３２に対するペアによって定義されている。例えば、ユーザ役割東部操作員は、{ルーター管理，東部} および {セキュリティ管理、東部} という２つのアクセス規則を持つ。

【００５５】ユーザ役割アクセス制御は、ユーザが１つまたは複数のユーザ役割にログインしたり役割を動的に切り換えることを可能にする強力で広範囲なセキュリティ・モデルである。本発明の好ましい実施形態において、ユーザが見るものをフィルタリングしたりユーザがとるアクションについて実行時検査を行うためユーザ役割アクセス制御がネットワーク環境で実施される。このセキュリティ・モデルの機能は、柔軟な構成および実施形態を提供することによって、従来技術モデルが遭遇するドメイン・モデル・セキュリティのような既存の問題を解決する。

【００５６】本発明のユーザ役割アクセス制御の実施形態が、ユーザが実行することができるアクションをＧＵＩ上でそれらアクションをフィルタリングすることによって決定されるという観点から主として記述されたが、ユーザ役割アクセス制御を別の形態で実施することが可能であることは理解されるべきである。ユーザが実行することができるアクションを"舞台裏で"フィルタリングすることによってアクションを決定するためユーザ役割アクセス制御を使用することもできる。この場合、ユーザがアクセス権を持つまたは持たない対象はユーザにとって透過的である。例えば、ユーザ役割は、ユーザがアクセス権を持つまたはそれに対してアクションを実行することができるマシンをフィルタリングすることによって、ユーザが実行することができるアクションを決定することもできる。この例では、ユーザ役割は、ユーザが、マシン１およびマシン２上に存在するがマシン３上には存在しないデータ・ファイルに対するアクセス権をユーザが持つことを決定することができる。ユーザがマシン１およびマシン２上のデータ・ファイルへのアクセス権を有してはいるが、それらはユーザが見ることができるようにＧＵＩに表示されず、むしろ、"舞台裏で"１つまたは複数の活動的ユーザ役割によって決定される。

【００５７】以上、本発明の好ましい実施形態を詳細に記述したが、本発明の概念は上記と異なる種々の形態で実施されることができる点は理解されるべきであろう。

【００５８】本発明には、例として次のような実施様態が含まれる。

(1)コンピュータ・オブジェクトに対するユーザのアクセスを制御する装置であって、該装置が、１つまたは複数のコンピュータ読み取り可能記憶媒体と、上記１つまたは複数のコンピュータ読み取り可能記憶媒体に所在し

て１つまたは複数のユーザ役割を作成するプログラム・コードと、を備え、上記１つまたは複数のユーザ役割の各々がユーザ・セットに関連づけられたアクセス権限セットを含み、上記アクセス権限の各々が上記ユーザ・セットの中のユーザによって実行される上記コンピュータ・オブジェクトに対するアクションを記述する、ユーザ・アクセス制御装置。

【００５９】(2)該装置が上記１つまたは複数のコンピュータ読み取り可能記憶媒体に所在する第２のプログラム・コードを更に含み、該第２のプログラム・コードが、所与のユーザがコンピュータ・セッションを初期化するたび毎に上記１つまたは複数のユーザ役割のどれが活動的であり該所与のユーザを参照するかを決定するステップと、活動的で所与のユーザを参照する上記１つまたは複数のユーザ役割に基づき、また、上記１つまたは複数のユーザ役割において所与のユーザに関連づけられているすべてのアクセス権限によって決定されているものとして、上記ユーザ・セットの中のユーザによって取られる上記コンピュータ・オブジェクトのセットに対する上記アクションをフィルタリングするステップと、を含む、前記(1)に記載の装置。

(3)上記ユーザ・セットの中のユーザによって取られる上記コンピュータ・オブジェクトのセットに対する上記アクションをフィルタリングする上記ステップが、所与のユーザがアクセスを許されているエレメントだけを該所与のユーザに活動的に表示するように、該所与のユーザに提示される１つまたは複数のグラフィカル・ユーザ・インタフェースのエレメントをフィルタリングするステップを含む、前記(2)に記載の装置。

【００６０】(4)該装置が上記１つまたは複数のコンピュータ読み取り可能記憶媒体に所在する第３のプログラム・コードを更に含み、該第３のプログラム・コードが所与のユーザがアプリケーションを起動するたび毎に上記１つまたは複数のユーザ役割のどれが活動的であって該所与のユーザを参照するかを決定するステップと、活動的で所与のユーザを参照する上記１つまたは複数のユーザ役割に基づき、また、上記所与のユーザに関連づけられているすべてのアクセス権限によって決定されているものとして、上記ユーザ・セットの中のユーザによって取られる上記コンピュータ・オブジェクトのセットに対する上記アクションをフィルタリングするステップと、を含む、前記(1)に記載の装置。

(5)上記ユーザ・セットの中のユーザによって取られる上記コンピュータ・オブジェクトのセットに対する上記アクションをフィルタリングする上記ステップが、所与のユーザがアクセスを許されているエレメントだけを該所与のユーザに活動的に表示するように、該所与のユーザに提示される１つまたは複数のグラフィカル・ユーザ・インタフェースのエレメントをフィルタリングするステップを含む、前記(4)に記載の装置。

【００６１】(6)該装置が上記１つまたは複数のコンピュータ読み取り可能記憶媒体に所在する第４のプログラム・コードを更に含み、該第４のプログラム・コードが、所与のユーザが所与のコンピュータ・オブジェクトに対するアクションを実行しようとするたび毎に上記１つまたは複数のユーザ役割のうちの１つの活動的ユーザ役割において上記所与のユーザと上記所与のコンピュータ・オブジェクトが関連づけられていることを検証するステップと、上記１つまたは複数のユーザ役割のうちの１つの活動的ユーザ役割において上記所与のユーザと上記所与のコンピュータ・オブジェクトが関連づけられている時にのみ上記所与のコンピュータ・オブジェクトに対するアクセス権限を上記所与のユーザに付与するステップと、を含む、前記(1)に記載の装置。

【００６２】(7)上記アクションのうちの所与のものがウェブ型環境におけるＵＲＬに対するアクセスを含む、前記(1)に記載の装置。

(8)上記１つまたは複数のアクセス権限の各々が１つまたは複数のアクセス規則によって決定され、上記１つまたは複数のアクセス規則の各々が操作セット／オブジェクト・ドメインのペアを含み、上記オブジェクト・ドメインがユーザ役割において参照されたユーザがそれに対するアクセス権限を持つ１つまたは複数のオブジェクトを含み、上記操作セットがユーザ役割において参照されたユーザが実行することができる１つまたは複数のアクションを含む、前記(1)に記載の装置。

【００６３】(9)コンピュータ・オブジェクトに対するユーザのアクセスを制御する方法であって、コンピュータ・セッションの間にアクションを実行する要求をユーザから受け取るステップと、上記要求に応答して上記ユーザがそのメンバとして属する１つまたは複数のユーザ役割を取り出すステップであって、上記１つまたは複数のユーザ役割の各々が、上記セッションに関して上記ユーザが有する１つまたは複数のアクセス権限を決定し、上記１つまたは複数のアクセス権限が上記セッションの間上記ユーザが実行することができる１つまたは複数のアクションを決定する、ステップと、上記ユーザがメンバである上記１つまたは複数のユーザ役割が上記要求されたアクションを許可するアクセス権限を含むか否かを決定することによって上記ユーザが上記要求されたアクションを実行する権限を与えられ手いるか否かを決定するステップと、を含む方法。

(10)上記１つまたは複数のアクセス権限が、ユーザがウェブ環境における１つまたは複数のＵＲＬにアクセスすることができるか否かを決定する、前記(9)に記載の方法。

【００６４】

【発明の効果】本発明のユーザ役割の利用によると、ユーザの特定のジョブに見合う権限を付与するユーザ・アクセス制御モデルが実現される。

【図面の簡単な説明】

【図１】ユーザ・アクセス制御のドメイン・モデルを示すブロック図である。

【図２】本発明において実施されるアクセス制御のユーザ役割モデルを示すブロック図である。

【図３】セッション・ログインに関連するコンポーネントを示すブロック図である。

【図４】セッション特性にアクセスするために使用されるJavaクラスを定義するサンプル・コードである。

【図５】セッション特性にアクセスするために使用されるＣ言語ＡＰＩ定義のサンプル・コードである。

【図６】ランチャ登録ファイルのサンプルを示すブロック図である。

【図７】ランチャ初期設定を通してＧＵＩをフィルタリングすることによってウェブ・ブラウザにおいて実施されるユーザ役割アクセス制御に関連するコンポーネントを示すブロック図である。

【図８】ＵＲＬ実行時にovlaunchコンポーネントを使用して実行時アクセス検査を行うことによってウェブ・ブラウザにおいて実施されるユーザ役割アクセス制御に関連するコンポーネントを示すブロック図である。

【図９】ＵＲＬ実行時にovlaunchコンポーネントを使用せずに実行時アクセス検査を行うことによってウェブ・ブラウザにおいて実施されるユーザ役割アクセス制御に関連するコンポーネントを示すブロック図である。

【図１０】本発明の第２の好ましい実施形態に関するセキュリティ構成の例を示すブロック図である。

【図１１】本発明の第２の好ましい実施形態に関するセキュリティ・モデルのサンプルを示すブロック図である。

【符号の説明】

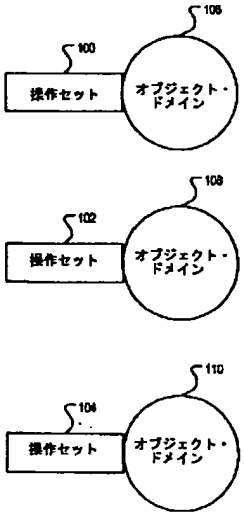
２００、２０２、２０４　　ユーザ役割

２０６、２０８、２１０、２１２、２１４、２１６、２１８、２２０、２２２　　アクション

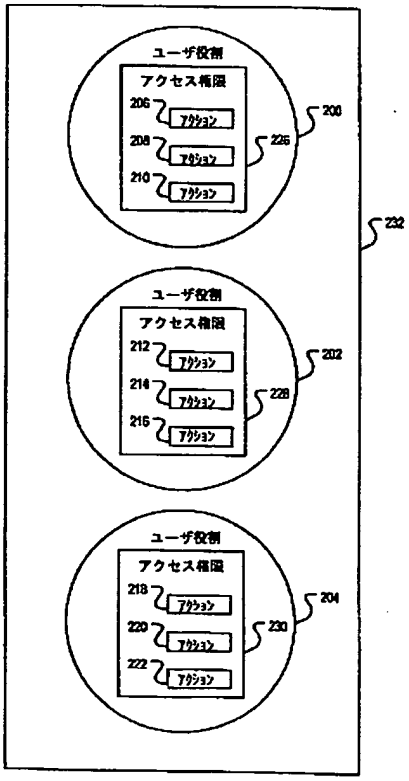
２２６、２２８、２３０　　アクセス権限

２３２　　コンピュータ読み取り可能記憶媒体

【図１】



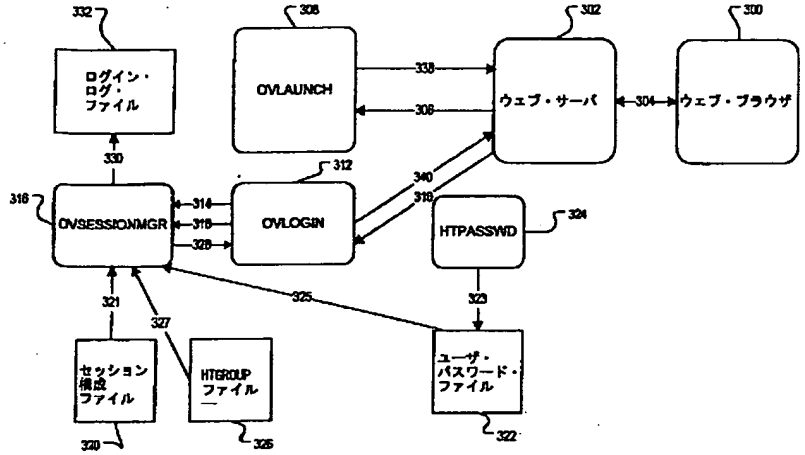
【図２】



【図６】

```
Application "Sample Web Application"
{
        DisplayString "Sample_Web_Application";
        Version "Release 8.02.00 May 19 1997";
        Copyright {
                "Copyright © 1997 No Company",
                "All rights reserved."
        }
        Description {
                "Just a sample registration file."
        }                                                          } 600
        Tab <100> "Tasks" "category\tasks.gif" ActiveHelp { "Tasks" }
        {
                <75> "Some Task" Icon "task.gif"
                        ActiveHelp {
                                "This is a brief description of
                                the task."
                        }
                        f.action "SomeTask";
                <80> "The Same Task" f.action "SomeTask";
                <45> "More Tasks" Icon "tasks.gif"
                        ActiveHelp {
                                "These are some more tasks"
                        }                                          } 602
                        f.list "MoreTasks";
        }
        List "MoreTasks"
        {
                "Another Task" Icon "tasks/icon.gif"               } 604
                f.action "AnotherTask";
        }
        Action "SomeTask"
        {
                URL "/OvCgi/somescript";
                Access netop sysop;
                WebWindow "WindowName" {
                        Type limited;
                        Status on;
                        Width 400;
                        Height 400;
                }                                                  } 606(a)
        }
        Action "Another Task"
        {
                URL "/OvCgi/somescript?Param=Another";
                Access admin;                                      } 606(b)
        }
}
```

【図３】



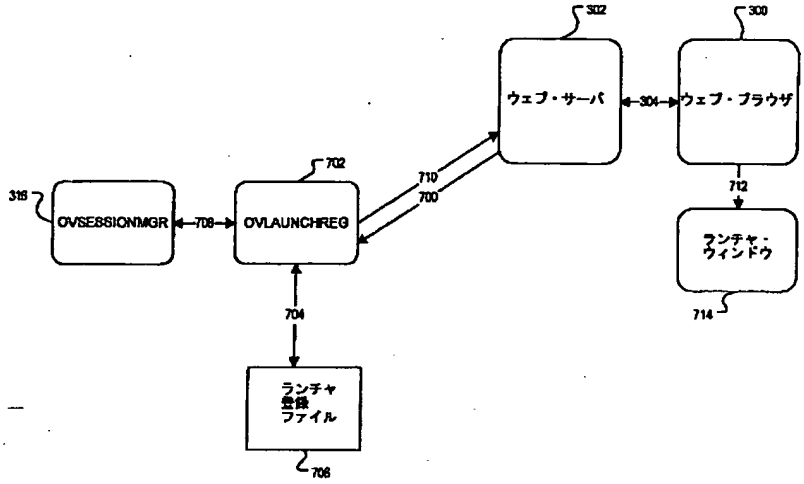
【図４】

```
package hp.ov.session

public class Session {
        // Initialize the session                                          } 400
        // Checks to make sure there is a valid session (i.e., the
        // user has logged on).  Creates the Java locale object.
        public Session(Applet applet) throws NoSessionException;

        public boolean isSessionValid();  402

        public boolean isLoginEnabled();  404

        // The session ID is hostname:n (e.g., "server.acme.com:1")        } 406
        public string getSessionID();

        // The user's login name (e.g., "cassandra")                       } 408
        // Returns NULL if user login is disabled
        public String getUser();

        // Blank-separated list of user roles (e.g., "netop sysadm")       } 410
        // Returns NULL if user login is disabled
        public String getRoles();

        // Verify access for the current user to a specified URL           } 412
        public boolean accessAllowed(string URL)
```

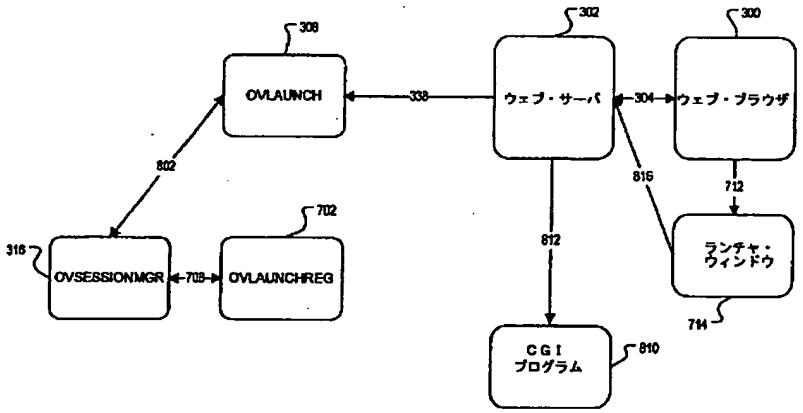
【図５】

```
#include "OV/ovweb.h"

/* Initialize the CGI program */                                           }
/* Sets LANG to the OS locale name and calls setlocale() */                }
/* If secured is TRUE, checks to see if the user is logged */              }
/* Brings up a login screen if the user is not logged in. */               } 500
/* CGIs that will not run in the context of a Web page should */           }
/* specify false for secured. */                                           }
/* If secured and no session exists, -1 is returned. */                    }
int OVwwwInit(boolean secured);

boolean OVwwwIsSessionValid();  502

/* The session ID is hostname:n (e.g., "server.acme.com:1) */              } 504
const char *OVwwwGetSessionID();

boolean OVwwwIsLoginEnabled();  506

/* The user's login name (e.g., "cassandra") */                            }
/* Returns NULL if user login is disabled */                               } 508
const char *OVwwwGetUser();

/* Blank-separated list of user roles (e.g., "netop sysadmin") */          }
/* Returns NULL if user login is disabled */                               } 510
const char *OVwwwGetRoles();

/* Check if user belongs to the role */                                    }
boolean OVwwwUserHasRole(const char *role);                                } 512

/* Verify access for the current user to a specified URL */                }
int OVwwwAccessAllowed(const char *url);                                   } 514
```

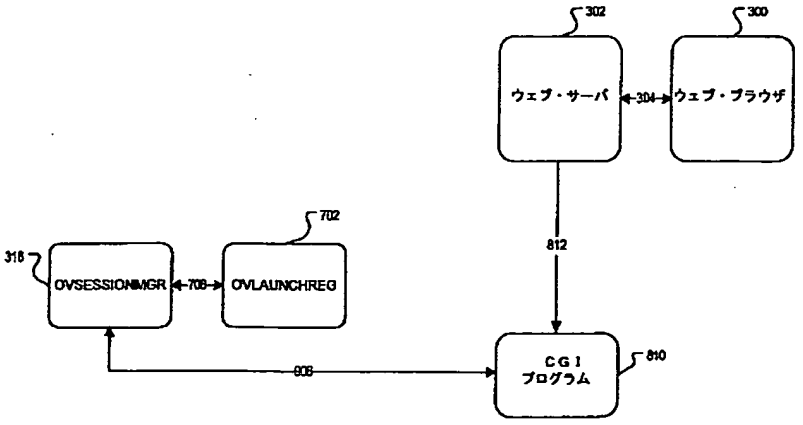
【図7】


302
300
ウェブ・サーバ
304
ウェブ・ブラウザ
712
ランチャ・ウィンドウ
714
316
OVSESSIONMGR
708
702
OVLAUNCHREG
710
700
704
ランチャ登録ファイル
706

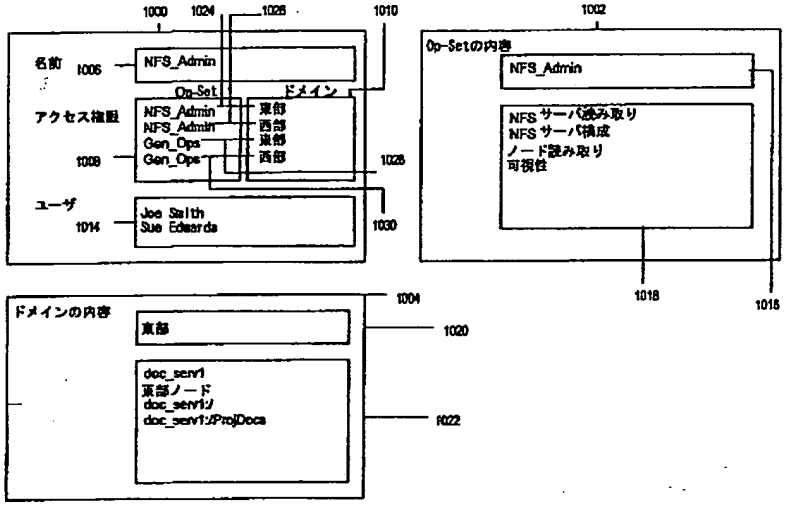
【図8】


306
OVLAUNCH
338
302
ウェブ・サーバ
304
300
ウェブ・ブラウザ
802
816
712
812
316
OVSESSIONMGR
708
702
OVLAUNCHREG
ランチャ・ウィンドウ
714
CGI プログラム
810

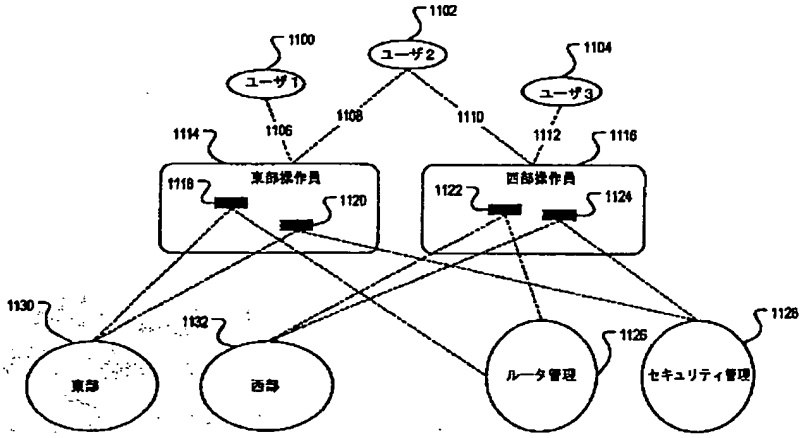
【図9】


302
300
ウェブ・サーバ
304
ウェブ・ブラウザ
812
316
OVSESSIONMGR
708
702
OVLAUNCHREG
908
CGI プログラム
810

【図１０】



【図１１】



フロントページの続き

(72)発明者　ブルース・シー・ウェルチ
アメリカ合衆国98004ワシントン州ベルヴュー、ノースイースト・ファースト・ストリート　9563

(72)発明者　ジュディス・シー・ウォーカー
アメリカ合衆国80525コロラド州フォート・コリンズ、サウスショアー・コート
4270